VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 Jakub Kadela

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Kadela**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Shopsys s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

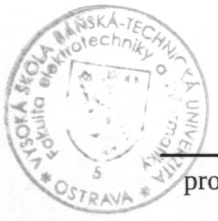
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Konzultant bakalářské práce: Ing. Václav Macíček

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 23. dubna 2019 ..... Kadeřa .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 23. dubna 2019

Rád bych na tomto místě poděkoval firmě Shopsys s.r.o. za příležitost vypracování bakalářské práce a všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Tato bakalářská práce popisuje odbornou praxi u firmy Shopsys s.r.o. Během této práce jsem se seznámil s agilním vývojem e-shopu a mnoha moderními technologiemi. Mým hlavním úkolem byla implementace nového e-shopu Růžový slon od počátku až po spuštění projektu do ostrého provozu. Webová aplikace je postavena na nejmodernějších webových technologiích stavěných na programovacím jazyce PHP. Mezi tyto technologie lze řadit například Symfony Framework, databázové technologie Doctrine nebo systém správ šablon Twig. Balalářská práce popisuje celý proces vývoje, od zaučení do Shopsys Frameworku, přes implementaci požadavků zákazníka až po finální přípravu před spuštěním.

Klíčová slova: e-shop, Shopsys, agilní vývoj, PHP, Symfony Framework, praxe

Abstract

This bachelor thesis describes professional practice in company Shopsys s.r.o. During this work I was introduced to agile development of e-shop and many modern technologies. My main task was implementation of the new e-shop Růžový slon from the very beginning to the end of the project. The web application is built on the latest web technologies built on the PHP programming language. These technologies include Symfony Framework, Doctrine database technology, and the Twig template management system. The bachelor thesis describes the whole process of development, from the training to the Shopsys Framework, through the implementation of customer requirements to the final preparation before launch.

Key Words: e-shop, Shopsys, agile development, PHP, Symfony Framework, practice

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
B2B	–	Business to Business
B2C	–	Business Customer
CI	–	Continuous Integration
CRO	–	Conversion Rate Optimization
CSS	–	Cascading Style Sheets
CSV	–	Comma Separated Values
DQL	–	Doctrine Query Language
HTML	–	Hyper Text Markup Language
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Securre
JS	–	JavaScript
LESS	–	Leaner Style Sheets
ORM	–	Object-Relational Mapping
PHP	–	Hypertext Preprocessor
SQL	–	Structured Query Language
SSH	–	Secure Shell
URL	–	Uniform Resource Locator

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce popisuje průběh mé praxe u společnosti Shopsys s.r.o, která se zabývá vývojem velkých e-commerce řešení. Pro praxi jsem se rozhodl z toho důvodu, že jsem chtěl uplatnit své teoretické znalosti při vývoji reálného softwaru a vyzkoušet si celý proces vývoje e-shopu. Během praxe jsem si také mohl vyzkoušet práci v týmu několika programátorů, testera, kodéra a analytika pod vedením projektového manažera.

1.1 O společnosti Shopsys

Společnost Shopsys se už přes 16 let zabývá vývojem a správou předních českých e-shopů [1]. Během těchto let vyvinula svůj vlastní produkt - Shopsys Framework, na kterém v posledních letech implementuje středně velké a velké e-shopy na míru. Spolu s vývojem e-commerce platformy nabízí společnost také služby spojené s moderní webovou aplikací, jako např. tvorba designu, datová analytika, testování i výkonnostní marketing.

Společnost Shopsys má několik poboček v České republice, a to v Ostravě, Pardubicích a v Praze, ale i na Slovensku v Žilině. Mezi její klíčové hodnoty patří důvěra, spolupráce, inovace a přátelskost. Tyto firemní hodnoty se snaží ctít a dodržovat ve všech směrech. O této společnosti jsem se dozvěděl od mého kamaráda, který zde již pracuje.

Obrázek 1: Logo Shopsys s.r.o.

1.2 Shopsys Framework

Během posledních let Shopsys vytvořil svůj vlastní produkt - Shopsys Framework [2]. Jedná se o open-source e-commerce platformu připravenou pro stavbu velkých e-shopů in-house týmy i agenturami vyvíjející e-shopy. Obsahuje důležité B2B a B2C e-commerce funkce a infrastrukturu připravenou na vysokou úroveň škálování. Využívá nejmodernější technologie pro vývoj internetových aplikací, jako např Docker a Kubernetes. Platforma je postavena na Symfony Frameworku, jednom z nejlepších PHP frameworků pro komerční software. Nechybí ani kontrola kódu pomocí coding standardů, automatické pětiúrovňové testování a připravená infrastruktura CI.

V současné době je Shopsys Framework používán pro stavbu e-shopů s obraty v řádech stamilionů korun. Do budoucna je v plánu vytvořit module store, který díky připraveným modulům třetích stran pomůže e-shopům ušetřit ještě více času a financí. Jednoduše tak půjde do stabilní verze přidat určitý modul, např. platební bránu, věrnostní systém, atd.

Aktuální stabilní verze platformy je označená Shopsys Framework 7. Během blízké doby budou vycházet releasy, na které půjde snadno povýšit z této verze. Projekt, na kterém jsem se v průběhu této praxe podílel, je však postaven na Shopsys Frameworku 6, neboť v této době ještě nebyla vydána stabilní verze 7 vycházející z verze 6. Nebylo tedy ještě možné využít povyšovatelnosti a škálování, které nabízí již nová verze frameworku.

2 Použité technologie

2.1 PHP, PostgreSQL

PHP je široce používáný skriptovací jazyk vhodný pro vývoj webových aplikací [3]. Používá se na straně serveru a slouží ke generování HTML kódu stránky, jenž pak server odesílá do prohlížeče. Mezi hlavní výhody patří nezávislost na platformě a široké možnosti využití. Umí například pracovat s mnoha databázemi, lze pomocí něj generovat i upravovat grafiku, odesílat a přijímat emaily, ale také podporuje všechny důležité internetové protokoly. Pro vývoj ve firmě jsem používal verzi PHP 7.2.

PostgreSQL je výkonný objektově-relační databázový systém [4]. Využívá a rozšiřuje jazyk SQL v kombinaci s mnoha funkcemi, které bezpečně ukládají a rozšiřují nejsložitější pracovní úlohy. Svou silnou pověst si získal díky osvědčené architektuře, integritě dat, spolehlivosti i rozšířitelnosti. Tento databázový systém byl ve firmě využíván pro ukládání všech dat z e-shopu. Jednalo se tak například o produktový obsah, objednávky a jejich stavy, články, magazín, parametry, příznaky a mnoho dalšího.

2.2 GIT

GIT je komplexní systém pro verzování souborů [5]. Tento systém zaznamenává změny souborů v průběhu času a uživatel tak může kdykoli obnovit jejich konkrétní verzi. Dokáže také efektivně zobrazovat rozdíly, slučovat změny více uživatelů a mnoho dalšího.

S tímto nástrojem jsem pracoval na denní bázi a s jeho pomocí jsem řešil jakoukoliv změnu v kódu. V průběhu praxe jsem se GIT naučil používat a pochopil jsem, že se jedná o nepostradatelný nástroj v programování. Pro správu repozitářů jsme ve firmě používali Gitlab.

2.3 Symfony, Twig, Doctrine

Symfony je webový aplikační framework pro vývoj webových aplikací [6]. Je napsán v jazyce PHP a vychází z návrhového vzoru MVC. Na této platformě je postaveno mnoho významných projektů jako např. Magento, PrestaShop, Sylius, Spryker, ale také produkt naší firmy Shopsys Framework.

Mezi hlavní výhody tohoto frameworku patří široká sada komponent a objektově orientovaná architektura. Tato platforma vychází z návrhového vzoru MVC, tudíž je rozdělena do tří vrstev - řízení, logika, výstup. Řízení obstarávají takzvané *Kontrolery*, které předají parametry uživatele modelům, od kterých získají data. Tato data předá pohledům (šablonám), které je začlení do HTML kódu a zobrazí v prohlížeči. Další vrstvou jsou *Modely*, které obsahují logiku aplikace, jako např. práci s databází nebo výpočty. Výstup dat obstarávají *Pohledy*, které do HTML šablon umožňují vkládat data z PHP pomocí speciálních značek.

Twig je šablonovací systém, který umožňuje oddělit aplikační vrstvu od prezentační [7]. Díky tomu je možné veškerou logiku a výpočty provést v aplikační vrstvě pomocí modelů a výsledky už pouze zobrazit skrze šablonu Twigu. Další výhodou je zvýšení bezpečnosti aplikace za pomocí automatického escapování obsahu proměnných.

Doctrine je ORM, které zajišťuje mapování objektů na relační databázi [8]. Entity v ORM zachycují objekty z reálného světa a transformují je do tříd v programovacím jazyku. Jedna z klíčových vlastností Doctrine je psaní databázových dotazů použitím Doctrine Query Language (DQL), který vychází z ORM frameworku Hibernate určeným pro Javu.

2.4 JavaScript, jQuery

JavaScript je interpretovaný, objektově orientovaný jazyk, známý jako skriptovací jazyk pro webové stránky [9]. Jedná se o skriptovací jazyk založený na prototypu, který je dynamický a podporuje objektově orientované, imperativní a funkční programovací styly. JavaScript běží na klientské straně webu, který lze použít k programování chování webových stránek při výskytu události.

jQuery je nejrozšířenější JavaScriptová knihovna s jednoduchou syntaxí [10]. Hlavní výhodou použití této knihovny je, že řeší nekompatibilitu mezi prohlížeči. Také je mnohem snadnější ji používat oproti standartnímu JavaScriptu.

2.5 Docker, Composer

Docker je open source software, jehož cílem je poskytnout jednotné rozhraní pro izolaci aplikací do kontejnerů [11]. Umožňuje rychle sestavit aplikace z komponent a eliminuje problémy, které můžou vzniknout při odesílání kódu. Mezi hlavní výhody patří rychlé doručení aplikací a snadnější nasazování a škálování. Tuto technologii jsem využíval pro instalaci a běh aplikace Shopsys Framework. Server a všechny potřebné služby běží v prostředí Dockeru a není potřeba je nastavovat a konfigurovat.

Composer je balíčkovací manažer naprogramovaný v jazyce PHP [12]. Tento manažer využívá veřejných repozitářů k stahování nebo správě komponent daného projektu. Pomocí těchto repozitářů tak lze snadno aktualizovat vybranou komponentu Symfony projektu nebo ji vrátit ke starší verzi. Veškeré údaje o stažených balíčcích si Composer ukládá do svých speciálních souborů *composer.lock* a *composer.json*, díky kterým pak může snadno provádět potřebné operace s těmito balíčky.

2.6 PhpStorm, Yodiz

Pro vývoj v Symfony frameworku jsem se rozhodl používat **PhpStorm**. Jedná se o produkt značky JetBrains a jako mnoho jiných vývojových prostředí nabízí možnost inteligentního našeptávače pro usnadnění práce [14]. Dále také nabízí komplexní prostředí pro vývoj, mnoho možností konfigurace a přidávání zásuvných modulů. Podporuje tvorbu webových aplikací v Symfony frameworku, poskytuje našeptávač funkcí pro tento framework a detekci anotací. Také má zabudovaný verzovací systém GIT i systém pro připojení k databázi. Zároveň jsem si ho vybral z důvodu, že toto prostředí používá většina zaměstnanců firmy.

Obrázek 2: Vývojové prostředí PhpStorm

Společně s PhpStormem jsem používal také nástroj **Yodiz**, který slouží pro agilní vývoj a plánování [15]. Pomáhá organizovat jednotlivé projekty a zároveň interaktivně zachytit celý proces agilního vývoje. V tomto nástroji jsem evidoval všechny úpravy, které jsem měl naplánovány k naprogramování a měnil úpravám stavy podle toho, v jaké fázi implementace jsem se zrovna nacházel. Také jsem pomocí tohoto nástroje měřil strávený čas na jednotlivých úkolech, aby se mohl vyfakturovat zákazníkovi.

3 Zaučení ve firmě

3.1 Pracovní pozice

Tato kapitola se bude zabývat mým nástupem do firmy, postupným zaškolením do Shopsys Frameworku a do ostatních potřebných technologií. Do společnosti Shopsys jsem nastoupil na pozici PHP Developer do týmu nových zakázek, který se zabývá implementací nových projektů. Tento celý tým je rozdělen do podtýmů, kde se každý zaměřuje na vývoj jiného e-shopu. Po mém nástupu mi byl přidělen nový projekt, čímž jsem si mohl projít celým procesem vývoje nové platformy.

3.2 Školení Frameworku

Při mém nástupu jsem absolvoval školení o firemních procesech, kde jsem se dozvěděl o fungování celé firmy. Po seznámení se s procesy jsem byl důkladně proveden funkčnostmi stabilního Shopsys Frameworku, na kterém se poté staví jednotlivé projekty. Následně jsem si prošel zdrojové kódy firemního frameworku a s pomocí kolegy, který mě zaučoval, jsem byl proveden jednotlivými částmi této platformy.

3.3 KinderGarten

Po úvodních školeních jsem se pustil do prvních testovacích úprav na framework platformě. Tomuto procesu se ve firmě říká KinderGarten (mateřská školka), kdy si programátor zkouší prvotní úpravy podle předem vymyšleného zadání. Tyto úkoly jsou navrženy tak, aby si daný člověk vyzkoušel co nejpestřejší úpravy napříč velkou částí Shopsys Frameworku. Po absolvování tohoto kurzu už jen zbývalo školení agilních metod a mohl jsem přejít na implementaci reálného projektu.

4 Agilní metody

Abych se mohl plně zapojit do týmu, který realizuje nový projekt, musel jsem se seznámit s agilními metodami vývoje softwaru používanými ve firmě.

4.1 Hlavní principy

Základním stavebním kamenem agilních metodik je tzv. *manifest agilního vývoje softwaru* [16]. Tento manifest říká:

- Jednotlivci a interakce před procesy a nástroji
- Fungující software před vyčerpávající dokumentací
- Spolupráce se zákazníkem před vyjednáváním o smlouvě
- Reagování na změny před dodržováním plánu

4.2 Klíčové role týmu

Pro správné fungování agilního týmu je potřeba definovat tyto role:

Scrum Master - jeho cílem je vytvořit samostatný, efektivní a spokojený self-organized tým. Pomáhá týmu dosáhnout jeho cílů a odstraňuje problémy. Motivuje ho k lepším výsledkům a chrání před vnějšími vlivy, které by ho mohly odvádět od soustředěné práce.

Product Owner - vlastník produktu. Má na starost definování vize projektu a její komunikaci týmu, zákazníkům i firmě. Rozhoduje, na které funkcionalitě se bude pracovat dříve. Většinu času tráví se zákazníkem, aby vstřebal jeho prostředí a dokázal vždy správně rozhodnout, kde je pravá hodnota zákazníka.

Self-organized team - tým musí rozumět zákazníkovi, chápat jeho prostředí a vědět, jak bude produkt používat. Musí jít všichni za stejným cílem a mít stejnou vizi. Tým by měl mít možnost rozhodovat a měnit to, jakým způsobem pracuje. Měl by táhnout za jeden provaz. Pokud selže jeden člen, selže celý tým.

Zákazník - zákazník je zapojen do projektu, aby si sám určoval priority a podílel se již v průběhu projektu na jeho změnách a funkcionalitě, aby se stal součástí týmu. V pravidelných iteracích ukazuje tým zákazníkovi dokončenou funkcionalitu.

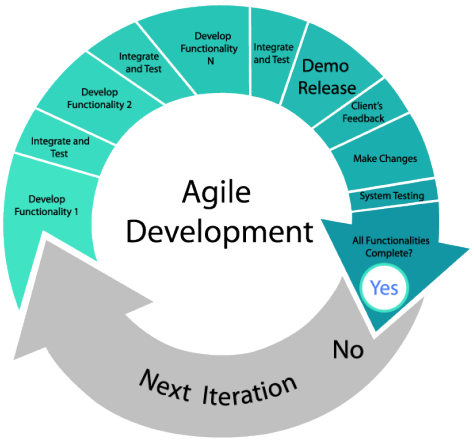
Project Manažer - obstarává všechny náležitosti tak, aby byl projekt správně reportován ve všech systémech firmy, stará se o vyplňování a sledování rozpočtových tabulek, případně o sledování času odpracovaného na produktu. Nerozděluje týmu úkoly a neudržuje projektový plán.

4.3 Agilní vývoj ve firmě

Celý vývojový proces e-shopu ve firmě je rozdělen na pravidelné cykly - Sprinty, ve kterých tým dodává hotovou funkcionalitu. Veškeré úpravy, které byly v tomto období naprogramovány, tým prezentuje na zákaznickém Demu vždy na konci Sprintu. Zákazník je tedy zapojen v průběhu celého vývoje do procesu a může rozhodovat o úpravách, které budou odbaveny v dalším období.

Pokud chce zákazník zařadit nějakou úpravu do dalšího Sprintu, sepíše požadavky a předá je Product Ownerovi, který tyto požadavky zpracuje a sepíše tzv. User Story - zadání od zákazníka. Tyto úpravy se poté naceňují programátory na Backlog Groomingu. Programátoři používají k ohodnocení jednotlivých User Stories relativní jednotky, podle kterých se dá jednoznačně určit, která funkcionalita je větší a která menší.

Po nacenění se úpravy připraví do fronty, aby se mohly v průběhu Sprintu odbavit. Jednotlivé úkoly musí projít fází implementace, přes *Code Review*, testování testerem až po business kontrolu Product Ownerem. Celý tento proces implementace je zachycen na tabuli, kde jsou jednotlivé User Stories zastoupeny papírovými odkazy, které jednotliví zaměstnanci posouvají dále, dokud se nedostanou do *Done*, do stavu, kdy je možné tyto úpravy na konci Sprintu předvést zákazníkovi.



Obrázek 3: Proces agilního vývoje

5 Projekt

Tato část bakalářské práce bude popisovat celý proces vývoje nového e-shopu, komunikaci se zákazníkem a spuštění projektu.

5.1 Růžový slon

Po mém zaškolení byl našemu týmu přidělen nový projekt Růžový slon. Jedná se o e-shop s erotickým zbožím, který si vybral naši firmu pro vývoj nové platformy. Zákazník měl už zastaralý a špatně udržitelný e-shop, který nedostačoval jeho růstu. Z tohoto důvodu si vybral Shopsys Framework, jakožto platformu, na které chce vyvíjet své individuální řešení na míru. V této části rozeberu celý proces vývoje nového řešení a všechny s ním spojené fáze.

5.1.1 Analýza a specifikace

Jako u každého projektu, i zde proběhla důkladná analýza a nacenění projektu. Tento proces mělo na starost několik analytiků, kteří sepsali požadavky klienta do specifikace. V tomto dokumentu byly shromážděny všechny individuální úpravy, které chtěl zákazník implementovat. Poté proběhl odhad náročnosti projektu a nacenění počtu hodin potřebných k vývoji a vedení celého projektu. Tento odhad byl dále zaslán klientovi, který souhlasil s počtem hodin i cenou produktu. Jednalo se však o odhad, který se mohl kdykoliv během implementace změnit z důvodu navýšení počtu úprav ze strany zákazníka.

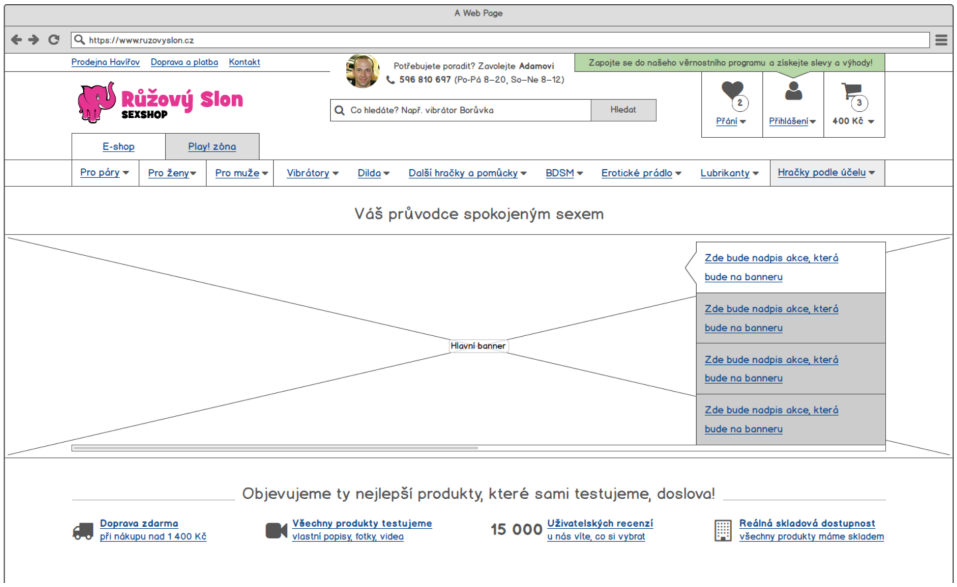
5.1.2 Grafika

Důležitou součástí vývoje platformy je také grafika a celkový vzhled nového e-shopu. Pro tento účel je ve firmě oddělení grafiků, kteří poskytují zákazníkovi služby pro vytvoření grafiky na míru. Pro prvotní vizualizaci e-shopu se používají wireframy, které slouží pro náhled nového řešení [17]. Wireframe definuje rozmístění funkčních prvků na stránce. Nejedná se v žádném případě o grafický návrh, neobsahuje obrázky a je tvořen pouze pomocí čar a textu. Tento způsob slouží pro jednoduchý náhled toho, jak by mohl výsledný produkt vypadat.

Pokud zákazník souhlasí s rozmístěním prvků na jednotlivých navržených stránkách, může se přejít k tvorbě grafických návrhů stránek e-shopu. Mezi první kroky patří vytvoření šablony barev a komponent (styleguide), která se používá při tvorbě výsledného vzhledu. Poté už nic nebrání vytvoření grafických vizuálů podle přání zákazníka, dle kterých se může následně kódovat vzhled HTML šablon. Tyto grafické návrhy se ukládají do nástroje Invision, který slouží pro správu těchto návrhů a komentování jednotlivých částí stránek. Pokud je zákazník schválí, může se podle těchto vizuálů začít kódovat vzhled stran e-shopu.

Ve firmě se používají dva způsoby implementace vzhledu, a to implementace čistých šablon před zahájením veškerých programátorských prací nebo nasazování vzhledu postupně při vývoji

funkčností. V prvním případě se nakódují všechny stránky a vzhled se nasadí ještě před spuštěním vývoje. Programátoři už poté jenom napojují vyvíjenou funkčnost na nasazený vzhled. Pokud je zvolena druhá varianta, začne se vyvíjet na stabilním vzhledu Shopsys Frameworku a zároveň s implementací úprav se postupně nasazuje grafika pro konkrétní naprogramované komponenty. U projektu, na kterém jsem se podílel, jsme využili druhou variantu.



Obrázek 4: Wireframe hlavní stránky webu Růžový slon

5.1.3 Počátek implementace

Jak už bylo zmíněno v předešlých kapitolách, v agilních metodách jsou zavedeny tzv. Sprinty, tudíž i zde to nebylo jinak. Product Owner tohoto projektu vybral po konzultaci se zákazníkem nejprioritnější věci a zařadil je do prvního Sprintu. Proběhl první Backlog Grooming, kde se sešel celý implementační tým, abychom mohli nacenit první úpravy. V kanceláři jsme také přivítali dva nové externí programátory z firmy Růžový slon, a to za účelem zaučení do Shopsys Frameworku. Po spuštění projektu bude jejich úkolem starat se o nový e-shop. Náš tým tedy s posilou dvou programátorů mohl začít s vývojem.

5.1.4 Implementace

V průběhu implementace jsme jako tým zodpovídali za to, že naplánované úpravy odbavíme do konce Sprintu. V našem případě trval Sprint přesně dva týdny. K odhadnutí náročnosti programovaných úprav slouží tzv. Story Pointy (relativní jednotky pro porovnávání náročnosti úprav). Dle těchto jednotek se dá jednoznačně určit, která úprava je náročnější. Nejedná se však o konkrétní počet hodin potřebných k práci.

Když je programátorovi přiřazen úkol, přesune se úprava do stavu *In progress*. Tento stav označuje, že programátor začal programovat a věnuje se této funkcionalitě. Pokud úkol dokončí, přesune úpravu na *Code review (CR)*. V této fázi si User Story může vzít jakýkoliv programátor, aby danou úpravu zkontroloval, zda-li je v pořádku naimplementovaná, nevyskytují se v ní bezpečnostní chyby a odpovídá všem zadaným standardům. Pokud je zde nalezen nějaký problém, je úprava vrácena zpátky programátorovi, který ji implementoval, a to k přepracování či opravě. Pokud je vše v pořádku, přesune se tato funkcionalita do další fáze a to na testování. Zde ji tester zkontroluje, zda odpovídá zadání a nevyskytují se v ní nějaké chyby. Následně je přenesena do předposledního stavu, a to na *Business validaci (BV)*. V této fázi úpravu kontroluje samotný Product Owner, který dohlíží nad tím, zda funkcionalita přináší zákazníkovi určitou business hodnotu a zda splňuje požadavky zákazníka. Pokud je i zde všechno v pořádku, je úprava dokončena a je přesunuta do stavu *Done*.

Všechny tyto stavy úprav jsou zaznamenávány v agilním nástroji Yodiz a současně fyzicky zrcadleny na tabuli v kanceláři. Každé ráno se koná *Standup*, kdy se celý tým sejde u této tabule a rozdělí si úkoly na celý den.

Obrázek 5: Tabule pro agilní vývoj

5.1.5 Testování

Mezi klíčové prvky vývoje patří také testování, které probíhá v několika fázích. První fází jsou automatické testy, které ověřují naprogramovanou úpravu při každé změně. Zdrojový kód je kontrolován mnoha testy, mezi které patří *Unit testy*, *Funkcionální testy*, *HTTP Smoke testy*, *Akceptační testy* a *Performance testy*.

Další fází po dokončení úpravy je testování týmovým testerem, který kontroluje, zda je úprava naprogramována přesně podle zadání a zda-li se v ní nevyskytují nějaké chyby. Také v průběhu testování poskytuje návrhy na zlepšení.

5.1.6 Demování

Po dokončení Sprintu je nutné naprogramované úpravy představit klientovi. K tomuto účelu slouží v agilním vývoji Demo, kdy se celý tým setká se zákazníkem a předvádí naprogramované funkčnosti. Demo tedy neslouží jen k zobrazení úprav ale i pro zpětnou vazbu od zákazníka. Jednotlivé připomínky se poté zapracují v následujícím Sprintu. Během vývoje projektu jsem si několikrát vyzkoušel předvedení úprav zákazníkovi. Byla to pro mě cenná zkušenost, jelikož jsem reálně konzultoval se zákazníkem naprogramované úpravy.

5.1.7 Spuštění

V průběhu vývoje e-shopu se tedy opakují jednotlivé Sprinty, dokud se neodbaví všechny zadané funkcionality. Poté přichází poslední fáze vývoje a to spuštění projektu. Této fázi předchází čas ladění posledních věcí a zapracování připomínek klienta. V průběhu celé implementace je také nutné vytvořit itinerář spuštění projektu a postupně ho doplňovat. Zde by se měly promítnout věci, které je nutno provést před spuštěním, během spuštění a po spuštění. Je například nutné mít přichystané HTTPS certifikáty na serveru, zkontrolovat všechny crony, změnit DNS záznamy po spuštění, smazat cache a zkontrolovat, zda vše funguje.

6 Zadané úkoly

Tato kapitola se bude zabývat již praktickou částí této práce a to implementací nového projektu podle specifikace zákazníka. V průběhu celého vývoje projektu jsem byl zodpovědný za několik desítek úprav na novém e-shopu. Pro prezentaci v této práci jsem tedy vybral ty nejzásadnější a nejkomplexnější. Mezi tyto klíčové úpravy bych zařadil implementaci platební brány Gopay, napojení na odběrná místa Zásilkovna, migraci dat ze starého e-shopu a implementaci progress barů pomocí CoffeeScriptu. V průběhu praxe jsem se dále podílel na úpravách jako implementace slevových kupónů, recenze produktů, indexovatelnost filtrů, úprava url ve filtru, tipy u produktů a také jsem byl zodpovědný za proces nasazení designu.

6.1 Zásilkovna

6.1.1 Zadání

Klient požadoval napojení na Zásilkovnu, což je síť výdejních míst v České republice i okolí [18]. Díky této úpravě si může zákazník vybrat v objednávce výdejní místo poblíž svého bydliště. Jedná se o nejrozšířenější českou přepravní společnost, poskytující komplexní logistické služby pro internetové obchody.

6.1.2 Popis funkce

Jednou za hodinu se pomocí cronu stahují všechny informace o pobočkách Zásilkovny a ukládají do databáze. Tato výdejní místa se poté zobrazují zákazníkovi v popup okně po kliknutí na dopravu typu Zásilkovna v objednávce na e-shopu. V tomto okně si může zákazník zadat město nebo PSČ do pole vyhledávání a podle tohoto výběru se mu zobrazí příslušná výdejní místa. U těchto míst vidí zakazník zobrazeny základní informace o pobočce a její otevírací době. Po zvolení výdejního místa už zákazník nevyplňuje dodací adresu, neboť je již předvyplněna ze zvolené pobočky Zásilkovny. Administrátor také vidí v administraci v detailu objednávky informace o vybraném výdejním místě.

6.1.3 Implementace

Před tím, než jsem začal s implementací této služby, vyžádal jsem si u zkušenějšího programátora z týmu předimplementační analýzu, jelikož jsem ještě nikdy nepracoval se službou třetích stran Zásilkovnou. Vysvětlil mi, jak tato služba funguje a jak bych ji měl naimplementovat.

Jako první krok jsem si vyhledal dokumentaci k implementaci Zásilkovny a zjistil jsem si, jaké atributy je nutno ukládat u jednotlivých poboček. Jednalo se o informace typu *id* pobočky, kompletní adresa, popis, GPS souřadnice aj. Následně jsem podle těchto zjištěných informací vytvořil celou modelovou část, tzn. entity pro uložení jednotlivých výdejních míst, entity pro práci s databází a entity pro práci s daty. Všechny tyto třídy jsem pojmenoval pomocí prefixu

PickUpPlace a umístil do adresáře v modelu aplikace. U plateb jsem si také přidal nový typ *Zásilkovna*, abych mohl rozeznat Zásilkovnu mezi ostatními druhy plateb. Dále jsem si v administraci zobrazil volbu tohoto typu a upravil šablonu pro detail platby v administraci. Poté, co jsem si vše připravil v modelové části a v šabloně v administraci, abych měl jednotlivá výdejní místa kde ukládat, jsem se mohl věnovat implementaci cronu pro stahování CSV souboru s pobočkami.

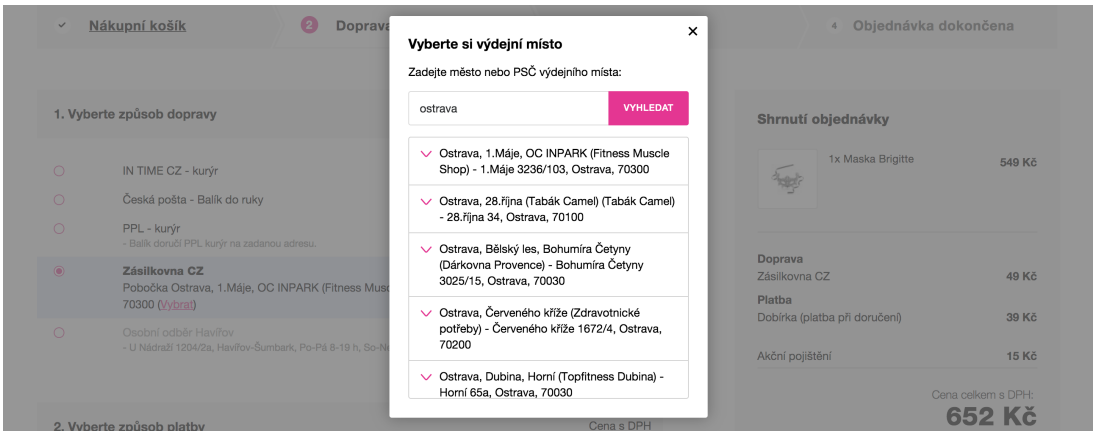
Ke stažení CSV souboru byl zapotřebí API klíč. Jednalo se o parametr, který se vkládá do URL adresy, aby bylo možné přistoupit k souboru výdejních míst. Pro účely vývoje jsme získali tento klíč k testovacímu CSV souboru s pobočkami Zásilkovny. Jelikož bylo zapotřebí, aby se tento klíč mohl po spuštění jednoduše nastavit přes SSH na serveru, rozhodl jsem se proto tento parametr umístit do konfiguračního souboru *parameters.yml*, kde půjde lehce zaměnit za ostrý klíč. Poté jsem mohl pokračovat s implementací cronu pro aktualizaci poboček. Vytvořil jsem si třídu pro stažení dat a rozparsování informací o výdejních místech Zásilkovny. Tato třída procházela jednotlivé řádky souboru, parsovala jednotlivé sloupce souboru a ukládala do objektu třídy *PickUpPlaceData*. Tento objekt byl následně uložen do databáze.

```php
/**
 * @param \SimpleXMLElement $xmlNode
 * @return \Shopsys\ShopBundle\Model\PickUpPlace\PickUpPlaceData
 */
private function parseZasilkovnaData(SimpleXMLElement $xmlNode)
{
    $pickUpPlaceData = new PickUpPlaceData();
    $pickUpPlaceData->placeId = (string)$xmlNode->id;
    $pickUpPlaceData->name = (string)$xmlNode->name;
    $pickUpPlaceData->street = (string)$xmlNode->street;
    $pickUpPlaceData->city = (string)$xmlNode->city;
    $pickUpPlaceData->description = $this->parseOpeningHours($xmlNode->
        openingHours->regular);
    $pickUpPlaceData->countryCode = $this->transformCountryCode((string)
        $xmlNode->country);
    $pickUpPlaceData->transportType = Transport::TYPE_ZASILKOVNA;
    $pickUpPlaceData->gpsLongitude = (string)$xmlNode->longitude;
    $pickUpPlaceData->gpsLatitude = (string)$xmlNode->latitude;
    $pickUpPlaceData->place = (string)$xmlNode->place;

    return $pickUpPlaceData;
}
```

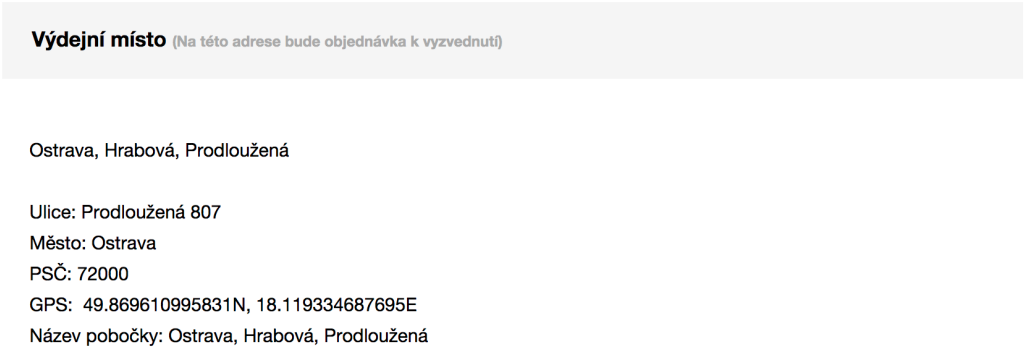
Výpis 1: Parsování řádku CSV souboru s pobočkami Zásilkovny

Podle toho, zda pobočka existovala v databázi, se vytvořila nová nebo editovala již stávající. Všechna data poboček byla uložena a mohl jsem začít se zobrazením na frontendu. V objednávce jsem pomocí JavaScriptu přidal popup okno pro vybrání pobočky se zobrazením otevírací doby. Také bylo nutné naprogramovat vyhledávací pole pro zadání PSČ nebo města. Po zadání hledané fráze byly pomocí AJAXU vyhledány pobočky, které odpovídaly výběru a následně vyrenderovány do šablony a zobrazeny uživateli.



Obrázek 6: Výběr výdejního místa v objednávce

Zbývalo mi tedy už jen vyplnit dodací adresu z vybraného výdejního místa v objednávce, aby ji měl zákazník již předvyplněnou a měl zjednodušenou práci. Pokud si tedy uživatel vybral dopravu typu Zásilkovna a zvolil konkrétní výdejní místo, byla mu adresa pobočky vygenerována jako doručovací adresa. Jako poslední bod jsem vytvořil demo data, která sloužila pro jednoduché otestování celé funkčnosti. Jednalo se tedy o vytvoření několika poboček, které se uložily do databáze již při buildu.



Obrázek 7: Zobrazení dodacích údajů po vybrání pobočky v objednávce

6.1.4 Problémy

Během implementace jsem nenarazil na žádné velké problémy. Veškeré otázky a nejasnosti jsem řešil s ostatními programátory v průběhu vývoje. Jediná věc, na kterou jsem v průběhu programování nemyslel, bylo to, že na obou doménách se zobrazovaly všechny pobočky. Po testování jsem ještě upravil zobrazení na jednotlivých doménách. Tzn. na české doméně se zobrazovaly pouze české pobočky a na slovenské pouze tamní pobočky.

6.1.5 Nabyté zkušenosti

V průběhu vývoje jsem se seznámil s vytvářením automatických cronů. Vyzkoušel jsem si parsování a získávání dat z CSV souboru. Také jsem se naučil pracovat se službou Zásilkovna a s její dokumentací.

6.2 Migrace dat

6.2.1 Zadání

Z důvodu převodu dat ze starého e-shopu bylo potřeba zmigrovat stará data na nový e-shop. Mým úkolem bylo převést data všech hodnocení a nahraných fotek k recenzím produktů. Programátor z firmy Růžový slon měl na starost vyexportovat potřebná data k migracím z databáze do dvou CSV souborů, abychom je mohli naimportovat do naší databáze. V jednom souboru byly uloženy všechny informace k recenzím a druhý soubor obsahoval pouze cesty k fotkám jednotlivých hodnocení.

6.2.2 Implementace

Jelikož jsem neměl zkušenosti s migrací dat, nechal jsem si před implementací poradit od jiného programátora a prošli jsme si možná řešení.

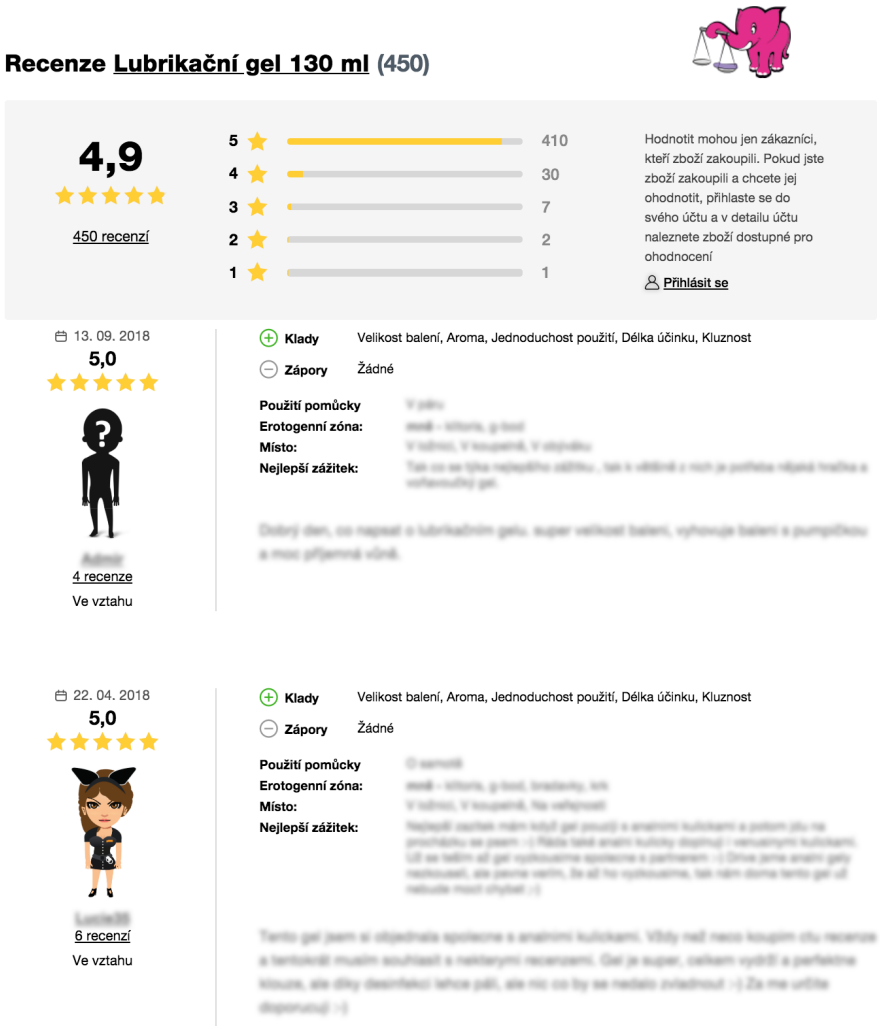
Zjistil jsem, že cílem bylo iterativně projít CSV soubor a naimportovat data do naší databáze. Pro tento účel bylo nutno naimplementovat *command*, který se spustí příkazem na serveru a který prochází celý soubor a načítá data. Soubor s recenzemi měl přes 20 000 záznamů, tudíž tento proces migrace trval okolo 15 minut. V souboru s cestami k fotkám se nacházelo už pouze 500 záznamů, které se nahrávály přibližně 4 minuty.

Nejdříve jsem vytvořil novou třídu *ImportProductReviewsFromCSVCommand*, která se bude starat o celý import dat. Tato třída rozšiřovala Symfony třídu *Command* [19], která obstarává práci s konzolí. Pomocí přetížení rodičovské funkce *configure()* je možné si nastavit název příkazu, kterým se *command* bude spouštět z terminálu na serveru, krátký popis a argumenty, např. cesty k souborům, ze kterých budeme číst data. V tomto případě jsem využil všechny tyto možnosti a nastavil *commandu* název, popis i cestu k souborům recenzí.

```
protected function configure()
{
    $this
        ->setName('shopsys:product:reviews-importfromcsv')
        ->setDescription('Import product reviews from given CSV file')
        ->addArgument(
            self::ARGUMENT_FILE,
            InputArgument::REQUIRED,
            'Source file review (*.csv)'
        );
}
```

Výpis 2: Konfigurace commandu v Symfony

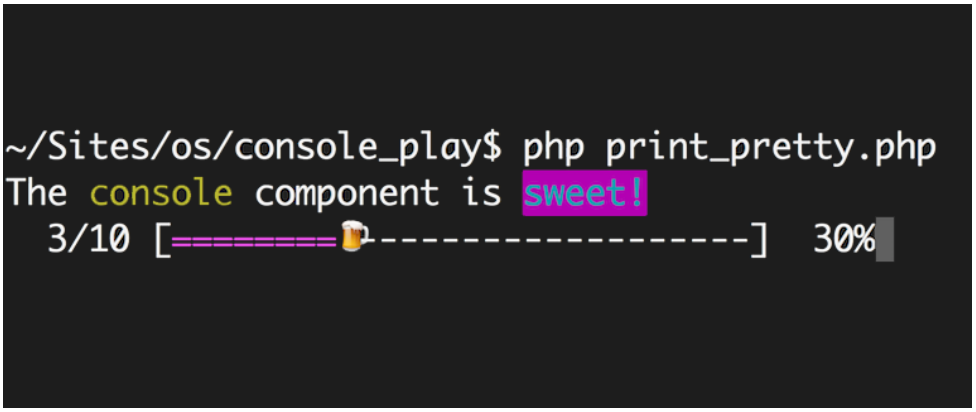
Poté už následovala samotná implementace funkce *execute()*, která se spustí příkazem, který jsme si v předchozím kroku nastavili. V této funkci jsem si načetl cestu souboru z nastaveného argumentu *commandu* a celý soubor načetl do paměti. Následně jsem pomocí cyklu procházel celé toto pole řádků souboru a jednotlivé sloupce ukládal do proměnných entity *ProductReviewData*. Z této datové entity jsem poté vytvořil objekt třídy *ProductReview* a tento objekt uložil do databáze. Tímto způsobem probíhala také implementace druhé třídy pro migraci obrázků k recenzím.



Obrázek 8: Zmigrovaná data recenzí na detailu produktu

6.2.3 Spuštění migrace

Po naprogramování celého commandu pro spuštění migrace a uložení dat jsem jej mohl spustit na mém lokálním notebooku. Jelikož Symfony podporuje v konzoli progress bary, rozhodl jsem se je naimplementovat i zde pro lepší vizuální kontrolu migrace. Také jsem vytvořil počítadlo naimportovaných recenzí, abych viděl, kolik jich ještě zbývá naimportovat a jak dlouho to bude trvat. Po několika minutách proběhly obě migrace dat a já jsem zkontroloval, zda se data v pořádku naimportovala na e-shop. Jelikož se recenze zobrazovaly na několika místech e-shopu, bylo nutné tato místa projít. Jednalo se o počítadlo recenzí nad patičkou webu, stránku se všemi recenzemi a detail produktu, kde se zobrazují hodnocení jednotlivého produktu.



Obrázek 9: Progress bar v konzoli Symfony

6.2.4 Problémy

Tato úprava nebyla nijak složitá, avšak v průběhu implementace jsem řešil několik problémů, u kterých jsem strávil nemálo času. Jednalo se hlavně o nekonzistenci dat v CSV souboru ze starého e-shopu. Musel jsem tedy řešit, že některé sloupce byly prázdné nebo obsahovaly data ve špatném formátu a tudíž jsem je musel upravit během migrace. Některé sloupce recenzí byly na starém e-shopu definovány jako nepovinné atributy, tudíž mohly obsahovat i *null*, ovšem na našem vyvíjeném e-shopu byly jako povinné.

Všechny tyto problémy jsem nakonec i s pomocí ostatních kolegů vyřešil. Bohužel některá stará data se při importu musela vynechat, protože zde chyběly opravdu důležité atributy. Celkově se tedy přeneslo asi 95% dat, což jsem považoval za úspěch.

6.2.5 Nabyté zkušenosti

Vzhledem k tomu, že jsem se dosud nesetkal s problematikou migrací, o to víc byla pro mě tato práce zajímavější a přínosnější. Naučil jsem se pracovat s *Commandem* v Symfony a s konzolí. Také jsem se zdokonalil v práci s CSV souborem.

6.3 Implementace progress barů

6.3.1 Zadání

Klient potřeboval naprogramovat v recenzním formuláři progress bary, které by zákazníkovi v reálném čase zobrazovaly stav vyplněnosti recenze. Zároveň by sloužily jako motivace pro získání slevového kupónu za vyplnění povinných polí. K získání výhodnějšího kupónu by musel zákazník vyplnit i nepovinná pole.

6.3.2 Předimplementační analýza

V této úpravě bylo zapotřebí rozhodnout, jakým způsobem bude řešeno naslouchání na formuláři, aby se zadané hodnoty kontrolovaly v reálném čase a zobrazoval se aktuální stav vyplněnosti v progress barech. Po předimplementační konzultaci s kolegou jsme usoudili, že pro tento případ není vhodné využít jQuery pro registrování změn na formuláři, neboť tato knihovna nepodporuje vytváření tříd a objektů. Vhodnější volbou byl CoffeeScript [20], který podporuje tvorbu tříd s jednoduchou dědičností.

6.3.3 Implementace

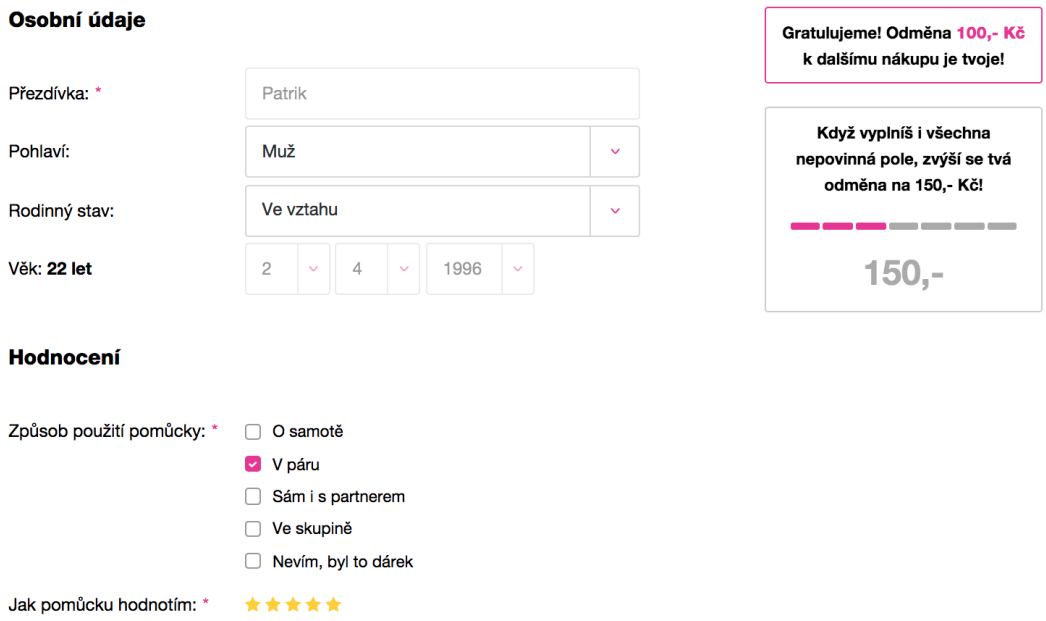
Začal jsem tedy s prací na tomto úkolu. Nejdříve bylo zapotřebí zajistit, aby se při každém buildu úpravy překompiloval CoffeeScript do JavaScriptu. Toho jsem docílil pomocí nástroje Grunt, který slouží pro automatizaci úkolů a v Shopsys Frameworku se využívá pro kompilaci Less [21] souborů do CSS [22]. Bylo tedy nutné doinstalovat rozšíření pro CoffeeScript do Gruntu pomocí balíčkovacího systému Composer. Následně jsem přidal nastavení do konfiguračního souboru, do tzv. Gruntfilu. Po otestování funkčnosti jsem začal s vytvářením souboru napsaného v CoffeeScriptu.

Nejprve jsem si vytvořil základní třídy pro práci s formulářovými prvky, jako je klasický input, textarea a select box. Tyto třídy jsem později využil jako základ pro implementaci pokročilých tříd, které zastupovaly jednotlivé prvky recenzního formuláře. Poté jsem si v šabloně formuláře k jednotlivým prvkům přidal unikátní CSS třídu, abych na těchto polích mohl registrovat eventy a mohl poznat, zda zákazník do těchto polí něco zadal a zda jsou data ve správném formátu.

Následně jsem si vytvořil třídu pro práci s progress bary, která zajišťovala všechnu práci s formulářem a na základě zadaných dat zobrazovala v progress barech aktuální stav vyplněnosti. Zde jsem vymyslel logiku vah pro jednotlivá pole a jak bude probíhat výpočet procent zobrazovaných v progress baru.

Nakonec jsem vyřešil serverovou část. Po odeslání se zkontrovaly vyplněné údaje a podle toho, zda uživatel vyplnil všechna povinná nebo dobrovolná pole, se vygeneroval příslušný kupón.

Tyto kódy měly mít také nastavené určité parametry. Mohly být využity pouze jednou, měly nastavenu minimální hodnotu objednávky na 150 Kč (6 Eur) a dobu platnosti na 1 rok.



Obrázek 10: Progress bary zobrazující vyplněnost recenzního formuláře

6.3.4 Problémy

V průběhu implementace této úpravy jsem narazil na několik problémů. Jelikož jsem nikdy nepracoval s CoffeeScriptem ani jiným podobným programovacím jazykem, musel jsem si hodně věcí nastudovat. Mnoho času jsem věnoval seznámení se s dokumentací a řešení problémů se syntaxí. Abych si danou část mohl vždy odzkoušet, musel jsem při každé změně překompilovat CoffeeScript soubor.

6.3.5 Nabyté zkušenosti

Jak už jsem zmínil v předešlé části, jelikož pro mě byla práce s CoffeeScriptem nová, měl jsem šanci se naučit pracovat s novou technologií a vyzkoušet si naprogramovat takto komplexní úpravu. Také jsem se seznámil s konfigurací Gruntu a kompilací CoffeeScriptu do JavaScriptu. V tomto programovacím jazyku jsem využil základní syntaxi, možnost rozšiřování tříd i využívání listenerů a naslouchání na prvcích formuláře.

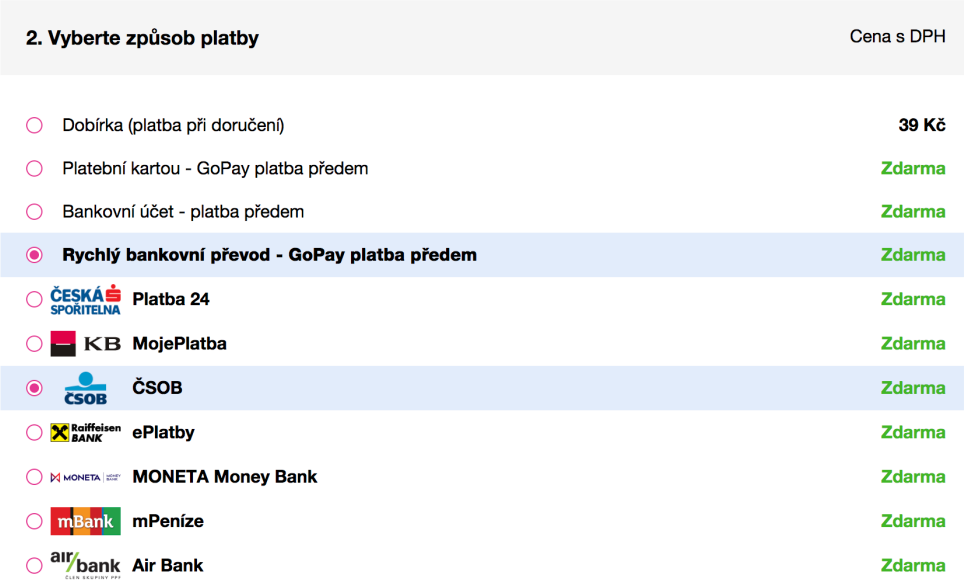
6.4 Platební metoda GoPay

6.4.1 Zadání

Klient požadoval naimplementovat platební bránu pro možnost placení kartou na novém e-shopu. Pro tento účel si vybral platební metodu GoPay, která patří mezi nejoblíbenější a nejkomplexnější řešení na trhu [23]. Mezi hlavní výhody patří inline platba, kdy zákazník zůstává po celou dobu na e-shopu a platební údaje vyplňuje v popup okně. Podporuje více než 55 platebních možností, jako platební karty, online bankovní tlačítka, bankovní převody, mobilní platby, elektronické peněženky i platby v bitcoinech. Dále umožňuje platit v 9 měnách s podporou 12 jazyků i opakované platby.

6.4.2 Popis funkce

Bylo zapotřebí vytvořit nový typ platby GoPay. Pokud si administátor vybere u platby tento typ, musí také rozhodnout, zda se jedná o platbu kartou nebo rychlý převod. Pokud zvolí *rychlý převod*, zákazník vidí v objednávce zobrazené jednotlivé bankovní platby spolu s bankovními převody. Po odeslání objednávky je přesměrován na stránku s online bránou, kde může zadat údaje potřebné k platbě.



Obrázek 11: Výběr bankovní platby a bankovního převodu v objednávce

Pokud se jedná o typ *platba kartou*, je zákazníkovi po odeslání objednávky zobrazen popup s platební bránou. Po úspěšném provedení platby je zákazník přesměrován zpátky na stránku s dokončenou objednávkou. V případě, že platba neproběhla korektně, je zákazník odkázán na stránku, kde je o tomto stavu informován.

6.4.3 Implementace

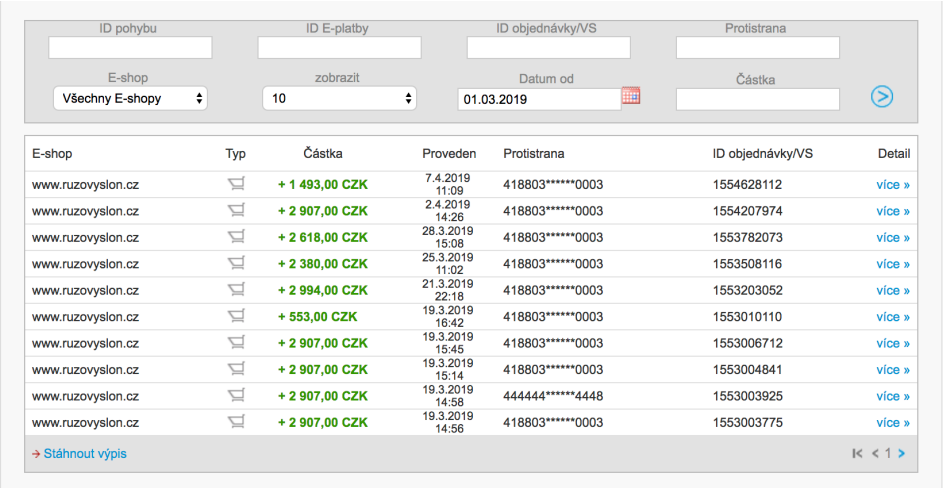
Před započetím prací jsem konzultoval tuto úpravu s programátorem, abych zjistil, jak funguje komunikace s platební bránou a jaký je proces celé platby. Po této předimplementační analýze jsem mohl začít s programováním.

Prvním krokem bylo přidání rozšíření *gopay* do balíčkovacího manažeru Composer. Tento balíček sloužil pro jednoduchou komunikaci s API platební brány Gopay. Jelikož se jednalo o multidoménový e-shop, bylo třeba mít pro každou doménu zvlášť platební bránu. Jako první jsem tedy přidal do konfiguračního souboru *parameters.yml* nastavení pro obě domény, kde se nastavovaly údaje jako *id* platebních bran a přihlašovací údaje. Pro účely testování jsem od Gopay dostal testovací brány pro obě domény i testovací prostředí, kde jsem si mohl později zkontrolovat všechny platby uskutečněné na e-shopu přes platební bránu a zjistit v jakém jsou stavu.

Dalšími kroky bylo vytvoření základní modelové části pro ukládání všech informací o platebních metodách, které se později budou stahovat cronem z API Gopay a implementace třídy pro komunikaci s API Gopay, která bude zajišťovat veškerou komunikaci mezi e-shopem a Gopay API. Také bude řešit stahování platebních metod, vracet statusy plateb a posílat platby objednávek do platební brány Gopay.

Poté jsem mohl začít s vývojem cronu pro stahování informací o platebních metodách. Tento cron jsem nakonfiguroval tak, aby se spustil co 4 hodiny a zaktualizoval potřebné informace. Dále bylo zapotřebí rozšířit objednávku, aby uchovávala informaci o vybrané platební metodě a o stavu platby. Posledním krokem bylo vytvoření stránky, která se má zobrazit, pokud se nepodaří uhradit objednávku nebo dojde k nějaké chybě. Zde byl zákazník vyzván k opakování platby. Tímto byla dokončena implementace platební brány a mohl jsem začít s testováním.

Pro ověření naprogramované funkčnosti jsem používal testovací karty, které jsem si našel v dokumentaci integrace Gopay [24]. Pomocí těchto platebních karet bylo odzkoušeno několik scénářů. Všechny proběhlé platby jsem si ověřil v testovacím prostředí Gopay *Sandbox*, kde jsou zobrazeny kompletní informace o platbách a finančním stavu internetového obchodu.



Obrázek 12: Testovací prostředí platební brány

6.4.4 Problémy

Během této implementace jsem narazil na několik problémů. Jelikož to byla první zkušenost s napojením na platební bránu, bylo zapotřebí si hodně informací vyhledat v dokumentaci a v průběhu implementace konzultovat s ostatními programátory. Dalším problémem, se kterým jsem se setkal, bylo odesílání emailů. Ve stabilní verzi frameworku je vygenerován email o dokončené objednávce hned při odeslání objednávky. Zde však bylo nutné změnit chování tak, aby se email odeslal až v případě uhrazení objednávky. Této doúpravě, na kterou mě upozornil tester, jsem věnoval více času, se kterým jsem však při implementaci nepočítal. Mezi další problémy bych zařadil i fakt, že některé scénáře plateb nebylo možné na lokálním počítači otestovat (například bylo zapotřebí mít nahraný HTTPS certifikát). Tyto scénáře bylo nutné ověřit po nasazení na testovací server.

6.4.5 Nabyté zkušenosti

Při vývoji platební brány jsem získal zkušenosti s napojením na Gopay i komunikací s API. Vyzkoušel jsem si práci s dokumentací třetí strany, bez které bych se při implementaci neobešel, a zdokonalil jsem se v navržení modelové architektury.

7 Závěr

Výsledkem mé práce ve firmě Shopsys jsou rozšíření pro Shopsys Framework naimplementovaná pro klienta přesně podle jeho požadavků. Tyto úpravy jsou součástí jednoho velkého celku a to nového e-shopu Růžový slon. Během této práce jsem se seznámil s celým procesem agilního vývoje projektu.

Tuto zkušenost s reálnou implementací bych zhodnotil velice pozitivně. Zdokonalil jsem se v návrhu architektury a napojení na služby třetích stran. Naučil jsem se nové programovací jazyky a také si vyzkoušel komunikaci s klientem.

Při vývoji projektu jsem uplatnil své dovednosti získané v průběhu studia. Využil jsem především znalosti architektur a návrhových vzorů z předmětu *Vývoj informačních systémů*, procesy softwarového vývoje z předmětu *Softwarové inženýrství*, ale i znalost objektově orientovaného programování z ostatních předmětů.

Zároveň bych chtěl poděkovat společnosti Shopsys za možnost vypracování bakalářské práce a také za nabyté zkušenosti v průběhu celé praxe. Rád bych vyzdvihl přátelskou atmosféru a týmovou spolupráci ve firmě, které jsou nezbytnou součástí úspěšného dokončení projektu.

Věřím, že znalosti a zkušenosti zde nabyté uplatním v budoucím studiu a později i v zaměstnání.

Literatura

[1] *Shopsys s.r.o* [online]. [cit. 2019-02-07]. Dostupné z: http://www.shopsys.cz

[2] *Shopsys Framework* [online]. [cit. 2019-02-07]. Dostupné z: https://www.shopsys.cz/co-je-shopsys-framework

[3] *PHP* [online]. [cit. 2019-03-14]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php

[4] *PostgreSQL* [online]. [cit. 2019-03-15]. Dostupné z: https://www.postgresql.org/

[5] *GIT* [online]. [cit. 2019-03-11]. Dostupné z: https://git-scm.com/

[6] *Symfony* [online]. [cit. 2019-03-15]. Dostupné z: https://symfony.com/

[7] *Twig* [online]. [cit. 2019-03-16]. Dostupné z: https://twig.symfony.com/

[8] *Doctrine* [online]. [cit. 2019-04-04]. Dostupné z: https://www.doctrine-project.org/

[9] *JavaScript* [online]. [cit. 2019-03-16]. Dostupné z: https://www.javascript.com/

[10] *JQuery* [online]. [cit. 2019-04-05]. Dostupné z: https://jquery.com/

[11] *Docker* [online]. [cit. 2019-03-12]. Dostupné z: https://www.docker.com/

[12] *Composer* [online]. [cit. 2019-03-14]. Dostupné z: https://getcomposer.org/

[13] *Grunt* [online]. [cit. 2019-03-22]. Dostupné z: https://gruntjs.com/

[14] *PhpStorm* [online]. [cit. 2019-03-22]. Dostupné z: https://www.phpstorm.com

[15] *Yodiz* [online]. [cit. 2019-03-22]. Dostupné z: https://yodiz.com/

[16] KUNCE, Eduard a Zuzana ŠOCHOVÁ. *Agilní metody řízení projektů*. Computer press, 2015. ISBN 978-80-251-4194-6.

[17] *Wireframe* [online]. [cit. 2019-04-05]. Dostupné z: https://cs.wikipedia.org/wiki/Wireframe

[18] *Zásilkovna* [online]. [cit. 2019-03-16]. Dostupné z: https://www.zasilkovna.cz/

[19] *Symfony Command* [online]. [cit. 2019-03-16]. Dostupné z: https://symfony.com/doc/3.4/console.html

[20] *Coffescript* [online]. [cit. 2019-03-25]. Dostupné z: https://coffeescript.org/

[21] *Less* [online]. [cit. 2019-04-11]. Dostupné z: http://lesscss.org/

[22] *What is CSS?* [online]. [cit. 2019-04-11]. Dostupné z: https://www.w3schools.com/css/css_intro.asp

[23] *GoPay* [online]. [cit. 2019-04-10]. Dostupné z: https://www.gopay.com

[24] *Gopay dokumentace* [online]. [cit. 2019-04-10]. Dostupné z: https://doc.gopay.com